unfolding Color Code の誤り耐性

1. 仮定

符号を用いているとき、エラーはシンドローム測定や誤り訂正中に発生しないと仮定する。もし仮にシンドローム測定中や誤り訂正中にエラーが起きたとするならば、それは次の round で訂正することとする。このことをわかりやすく、Fig.1 に示した。Fig.1 に示す通り、round 2 の操作時に発生するエラーは、round 2-round 3 間の論理操作時に発生するエラーと置き換えることができると仮定するということである。そのため、round t では round $t-1$-round t 間のエラーが訂正できれば良い。

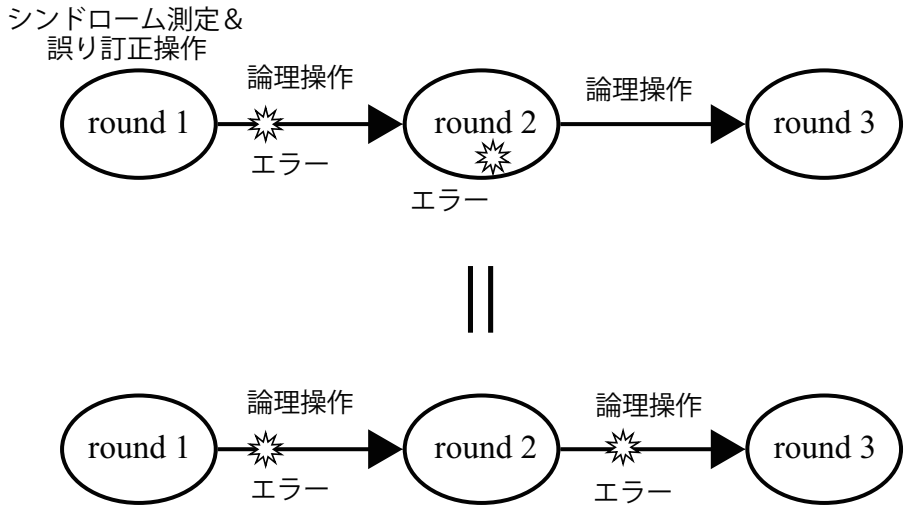

Fig. 1

このようなことを前提として以下では unfolding Color Code の誤り耐性、具体的には Color Code から Surface Code への変換（Fig.1 でいうとシンドローム測定）の際に発生したエラーがその変換後に検出できるのかということを確認していく。

2. エラー検出

まず、Stabilizer Code ではどのようにしてエラーを検出しているのかということを確認する。ここで、round t での physical qubits の状態は $|\psi\rangle^t$ と表され、round t ではエラーは発生していないとする。また、stabilizer group を $\mathcal{S}$ とする。round t でスタビライザー $s_j \in \mathcal{S}$ についてシンドローム測定を行うと、

$$s_j |\psi\rangle^t = m_j^t |\psi\rangle^t \tag{1}$$

が成り立つ。ただし、m_j^t は round t での s_j のシンドローム値を表す。スタビライザー符号ではよくスタビライザー状態を $m_j = 1$ として定義するが、実際にエラーを検出するときは $m = -1$ のときをスタビライザー状態としても同じであり、このときは $m_j = 1$ のときにエラーが発生していることになる。次に、round t-round $t+1$ 間で論理操作をし、round $t+1$ で $s_j \in \mathcal{S}$ についてシンドローム測定を行うと、

$$s_j |\psi\rangle^{t+1} = m_j^{t+1} |\psi\rangle^{t+1} \tag{2}$$

が成り立つ。このとき、round t-round $t+1$ 間で s_j によって検出されるようなエラーが発生していたら、m^{t+1} と m_j^t の符号は逆、すなわち $m^{t+1}m_j^t = -1$、round t-round $t+1$ 間で s_j によって検出されるようなエラーが発生していなければ、m_j^{t+1} と m_j^t の符号は同じ、すなわち $m^{t+1}m_j^t = 1$ となる。このようなことから、round t-round $t+1$ でのエラー検出はすべてのスタビライザーに対して、$m_j^{t+1}m_j^t$ を考えれば良く、$m_j^{t+1}m_j^t = -1$ のとき、エラーが起こったとする。

3. unfolding 操作でのエラー検出

Color Code の unfolding 操作は Fig.2 のようにして行われる（詳細は前回の資料 unfolding_color_code.pdf を参照）。

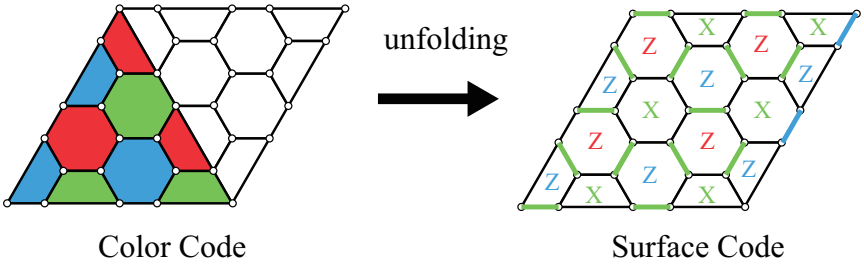

Fig. 2

ここでの議論は round t で Fig.2 の左の状態、round $t+1$ で Fig.2 の右の状態であるとする。また、符号距離は一般の d のときについて議論する。

まずもともと Color Code だった領域で unfolding 操作中に発生した X エラーに注目する（Fig.3）。

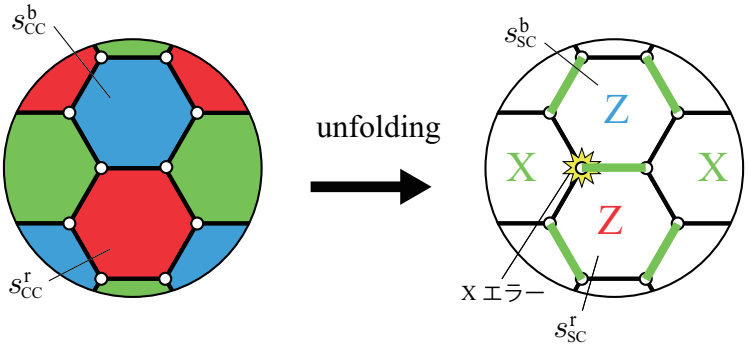

Fig. 3

Fig.3 で見えているスタビライザーについて、Color Code の blue face Z スタビライザー、red face Z スタビライザーをそれぞれ s^{b}_{CC}, s^{r}_{CC}、Surface Code の Z スタビライザーでもともと blue face だった部分、red face だった部分を s^{b}_{SC}, s^{r}_{SC} と表す。また、s^{i}_{j} のシンドローム値を m^{i}_{j} とする。このとき、Fig.3 に示されている X エラーというのは s^{b}_{SC}, s^{r}_{SC} のシンドローム値を反転させる。つまり、$m^{b}_{SC}m^{b}_{CC} = -1$, $m^{r}_{SC}m^{r}_{CC} = -1$ となり、発生した X エラーを検出できる。すなわち、Fig.4 のように、Surface Code 上では anyon model を用いて粒子 m が発生したのと同じである。あとは Surface Code と同じように誤り訂正すれば良い。Boudary 部分についても同じような議論ができる。また、もともと Color Code ではなかった部分の X エラーについても、最初は Z 方向に初期化されていることから、操作前のシンドローム値は初期化したときの 1 つの正六角形状の 6 つの qubit の測定値の積と、その場所の Surface Code のシンドローム値を比べることによって X エラーを検出できる。Bundary に 2-weight スタビライザーが存在する場合があるが、それも同じようにできる。

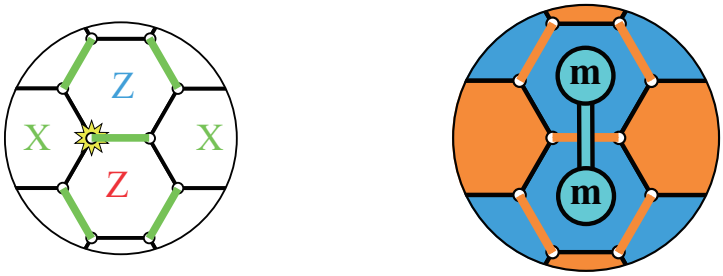

Fig. 4

次にもともと Color Code だった領域で unfolding 操作中に発生した Z エラーに注目する（Fig.5）。

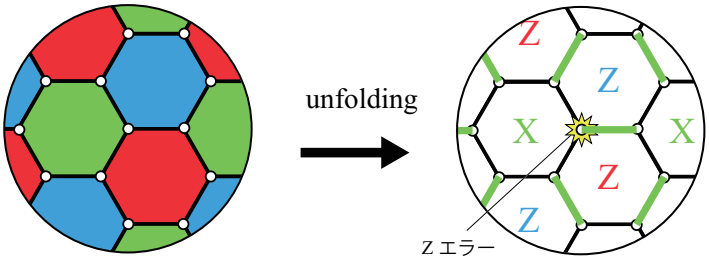

Fig. 5

unfolding 操作の 2-weight X スタビライザーは red face の Z スタビライザーと反可換であるため、いくつかの 2-weight X スタビライザーは undeterministic である。しかし、エラーが存在しなければ、1 つの blue face（red face）上の 3 つの 2-weight X スタビライザーの積は determinisic で Color Code のときの 6-weight X スタビライザーのシンドローム値と同じでなければならない。すなわち、Fig.5 の Z エラーは検出できる。また、anyon model で表すと Fig.6 となる。

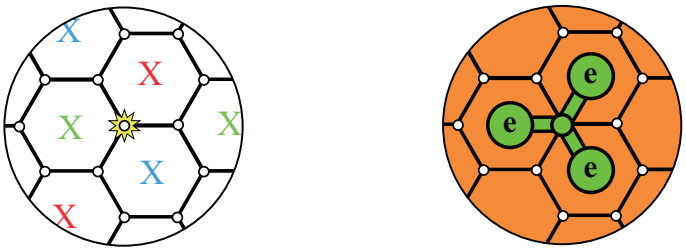

Fig. 6

エラーの位置がわかるので、これを Surface Code の描像に mapping すると Fig. 7 のようになる。また、$|0\rangle$ に初期化する領域には Z エラーが起きない（？）ので考える必要は無い。あとは Surface Code と同じように誤り訂正すれば良い。

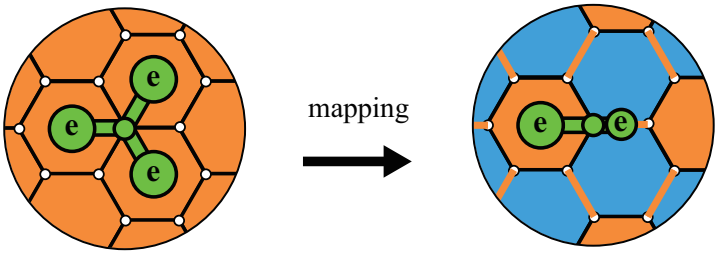

Fig. 7

4. mapping

実際に Surface Code の描像への mapping がうまくできるか検証してみる。

まず、X エラー検出は素直にできる（Fig.8）。ここで、黄色のチェインは X エラーを表す。

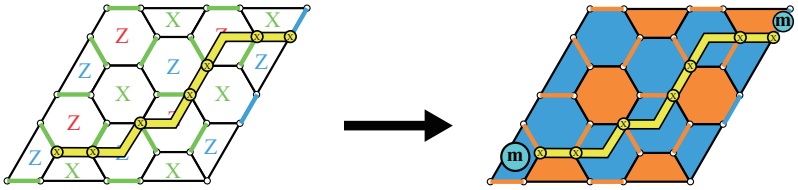

Fig. 8

次に、Z エラーの検出だが、これは少し工夫が必要である。Fig. 9、Fig. 10 に 2 パターン示す。ここで、黄色のチェインは Z エラーを表す。行っていることは、Surface Code の描像の anyon model で表せるまで、Fig. 7 の形を作って変換しているだけである。本当は右上半分には Z エラーは存在しない（？）が、存在しないほうが簡単なのであまり気にしなくても良い。

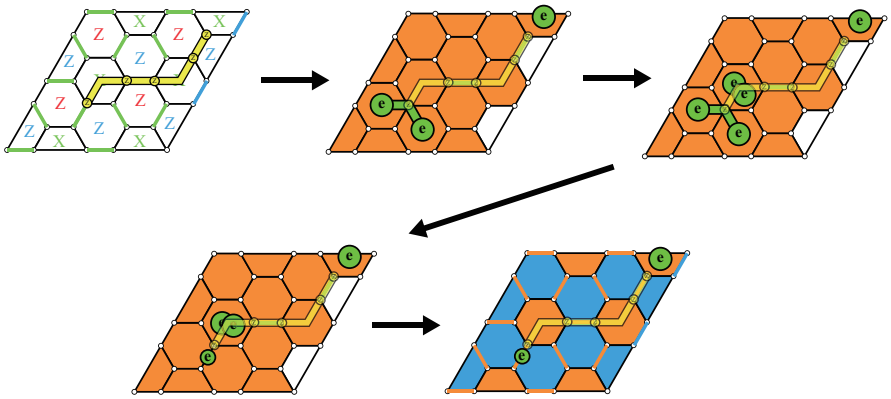

Fig. 9

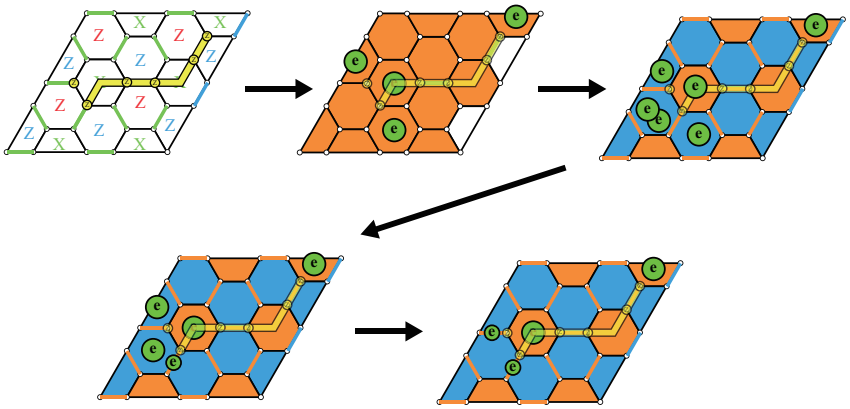

Fig. 10

あらゆるパターンをこのように変換できる（？）。